Claims:

1. A method of modifying program code stored in read only memory (ROM), comprising:

a) preceding each potentially modifiable code segment in ROM with a call to a lookup function which is logically external to the ROM;

b) storing the lookup function in a random access memory (RAM);

c) storing replacement code segments in the RAM with addresses;

d) providing the lookup function with an address table whereby the lookup function determines whether or not each potentially modifiable code segment in ROM has a replacement code segment in RAM; and

e) executing the replacement code segment in RAM in place of the potentially modifiable code segment in ROM whenever the lookup function determines that a potentially modifiable code segment in ROM has a replacement code segment in RAM.

2. A method according to claim 1, wherein:

each call to the lookup function includes a return address corresponding to the code segment it precedes.

3. A method according to claim 2, further comprising:

f) returning to the return address when the lookup function determines that there is no replacement code segment in RAM corresponding to the return address.

4. A method according to claim 2, wherein:

the return addresses are used to index the address table.

5. A method according to claim 1, wherein:

each replacement code segment includes a ROM address to which program execution will return after executing the replacement code segment.

6. A method according to claim 1, wherein:

said method operates independent of programming language.

7. A method according to claim 1, wherein:

the ROM is part of a computer peripheral, and

the RAM is loaded by the computer.

8. A method according to claim 1, wherein:

the RAM is loaded by a bootstrap program.

9. A method according to claim 7, wherein:

the computer peripheral is a modem.

10. An embedded system, comprising:

a) a processor;

b) a read only memory (ROM) coupled to said processor and containing program code for execution by said processor; and

c) a random access memory (RAM) coupled to said processor and containing at least one replacement code segment for replacing a segment of said program code in ROM and a lookup function, wherein

each potentially modifiable code segment in ROM is preceded with a call to said lookup function,

said at least one replacement code segment has an address,

said lookup function has an address table whereby the lookup function determines whether or not each potentially modifiable code segment in ROM has a replacement code segment in RAM, and

said processor executes the replacement code segment in RAM in place of the potentially modifiable code segment in ROM whenever the lookup function determines that a potentially modifiable code segment in ROM has a replacement code segment in RAM.

11. An embedded system according to claim 10, wherein:

each call to said lookup function includes a return address corresponding to the code segment it precedes.

12. An embedded system according to claim 11, wherein:

said processor returns to the return address when the lookup function determines that there is no replacement code segment in RAM corresponding to the return address.

13. An embedded system according to claim 11, wherein:

the return addresses are used to index the address table.

14. An embedded system according to claim 10, wherein:

each replacement code segment includes a ROM address to which program execution will return after executing the replacement code segment.

15. An embedded system according to claim 10, wherein:

said embedded system is part of a modem.

16. A method of modifying program code stored in read only memory (ROM), comprising:

a) preceding each potentially modifiable code segment in ROM with a call to a function which is logically external to the ROM;

b) storing the function(s) in a random access memory (RAM); and

c) executing the function in RAM when called by the calls in ROM.

17. A method according to claim 16, wherein:

said functions are chosen from the group consisting of a dummy function which returns execution to ROM and a fragment of replacement code which is executed and returns to ROM at an address subsequent to the address of the call which called the function.

18. A method according to claim 16, wherein:

the calls to functions are evenly spaced throughout the code in ROM.

19. A method according to claim 16, wherein:

said step of storing functions in RAM is executed at run-time.